Terraform Task

Name:Shaik Khaja Basha
Batch : Batch 11
Date : 22.07.2025
Task : datatypes

---

1. **Defines what kind of data a variable can hold?**

Ans: Terraform Data Types

Data types define what kind of data a variable can hold in Terraform.

Primitive Types

1. String

- Alpha-numeric values that can include symbols
- Enclosed in double quotes " "
- \n for multi-line strings

```
variable "example" {
  type    = string
  default = "test123"

  # Multi-line example
  default = "test123\nshgf"
}
```

2. Number

- Numeric values (both integer and decimal)
- No separate float/integer distinction

```
variable "example" {
  type    = number
  default = 125    # Integer
  default = 10.5   # Decimal
}
```

3. Boolean

- Conditional data type
- Only true or false values

```
variable "example" {
  type    = bool
  default = true
}
```

4. Any (Default)

- Accepts any data type
- Type is inferred from the default value

```
variable "example" {
  # type not specified defaults to 'any'
  default = 10      # becomes number
  default = "test"  # becomes string
  default = false   # becomes bool
}
```

Complex/Composite Types

1. List

- Ordered collection of similar values
- Can be nested (lists of lists)
- Accessed by index (0-based)

```
# Unconstrained list
variable "untyped_list" {
  type    = list
  default = ["test", 123, true, "test", 123]
}

# Typed list
variable "number_list" {
  type    = list(number)
  default = [1,2,3,4,5,2,4,7,1,2]
}

# Nested list
variable "nested_list" {
  type    = list(list(number))
  default = [[1,2],[3,4],[5,6]]
}

# Access: var.number_list[2] → 3
# Error if index out of bounds
```

2. Set (Partially Deprecated)

- Collection of unique values
- Automatically removes duplicates
- Order not guaranteed

```
variable "example_set" {
  type    = set(number)
  default = [1,2,3,4,5,2,4,7,1,2] # Stored as [1,2,3,4,5,7]
}
```

3. Map

- Key-value pairs
- Keys are always strings
- Values can be constrained

```
# Unconstrained map
variable "untyped_map" {
  type = map
  default = {
    name     = "adi"
    id       = 123
    isactive = true
  }
}

# Typed map
variable "string_map" {
  type = map(string)
  default = {
    name     = "adi"
    id       = "123"
    isactive = "yes"
  }
}

# Access: var.string_map["name"] → "adi"
# Error if key doesn't exist
```

4. Tuple

- Fixed-length sequence with specific types
- Position determines type

```
variable "network_config" {
  type = tuple([string, number, bool])
  default = ["192.168.1.0", 24, true]
}
```

5. Object

- User-defined structured type
- Each attribute has specific type
- Combines features of map and struct

```
variable "user" {
  type = object({
    name     = string
    age      = number
    active   = bool
    contacts = optional(list(string))
  })
  default = {
    name   = "John"
    age    = 30
    active = true
  }
}
```

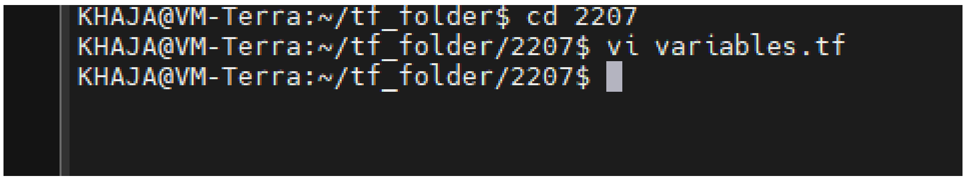
Type Injection via terraform.tfvars

```
# For lists
number_list = [1,2,5,6.7]

# For maps
user_map = {
  name = "test"
  dob  = 123
}
```

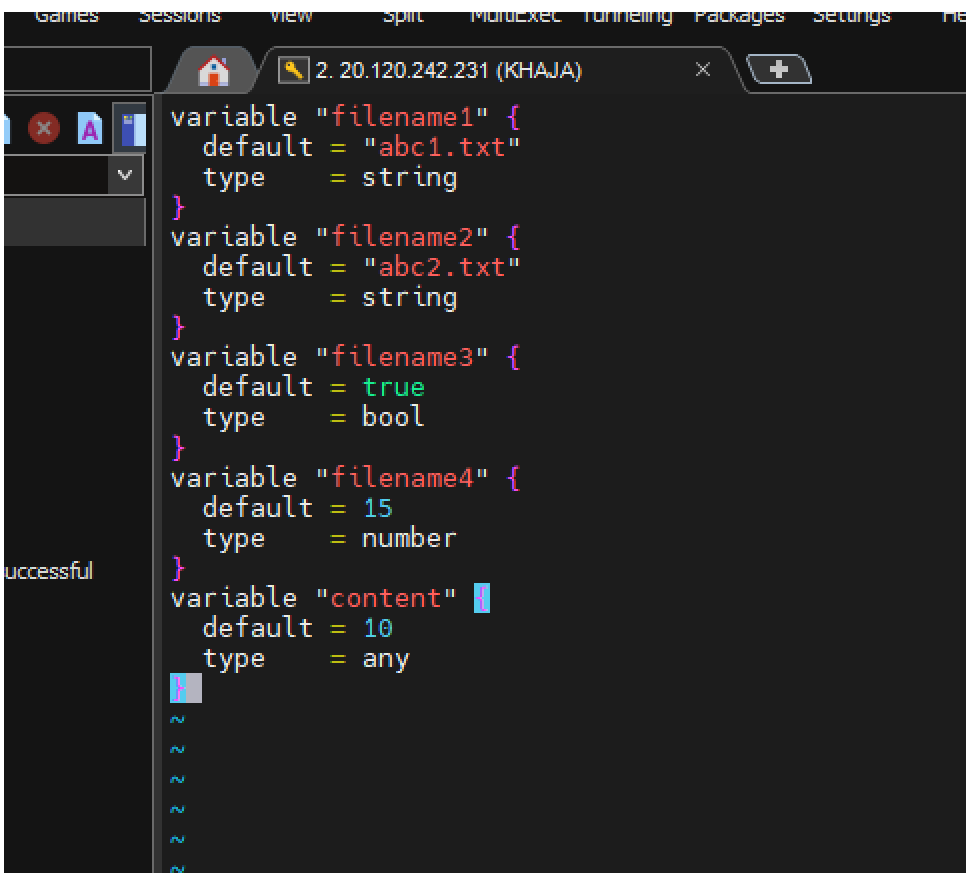
# Primitive Data Type Demo Using local_file Variables

```
# 1. String Variable
variable "filename1" {
  default = "abc1.txt"
  type    = string
}

# 2. String Variable (same as above, but explicitly typed)
variable "filename2" {
  default = "abc2.txt"
  type    = string
}

# 3. Bool Variable – it’s a Boolean type
variable "filename3" {
  default = true
  type    = bool
}
```

```
# 4. Number Variable
variable "filename4" {
  default = 15
  type    = number
}

# Content can be of any type
variable "content" {
  default = 10
  type    = any
}
```



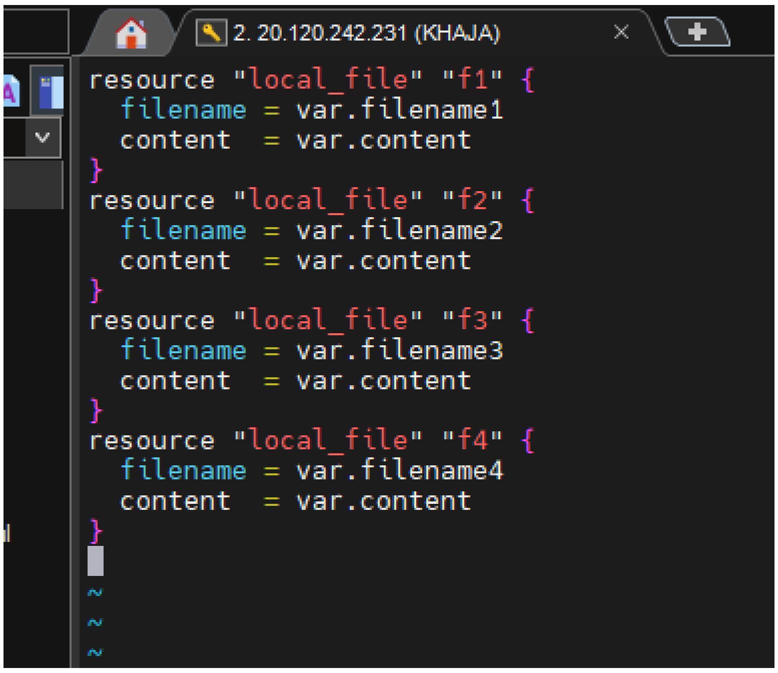
```
KHAJA@VM-Terra:~/tf_folder$ cd 2207
KHAJA@VM-Terra:~/tf_folder/2207$ vi variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$
```



```
variable "filename1" {
  default = "abc1.txt"
  type    = string
}
variable "filename2" {
  default = "abc2.txt"
  type    = string
}
variable "filename3" {
  default = true
  type    = bool
}
variable "filename4" {
  default = 15
  type    = number
}
variable "content" {
  default = 10
  type    = any
}
~
~
~
~
~
~
```

Resources are

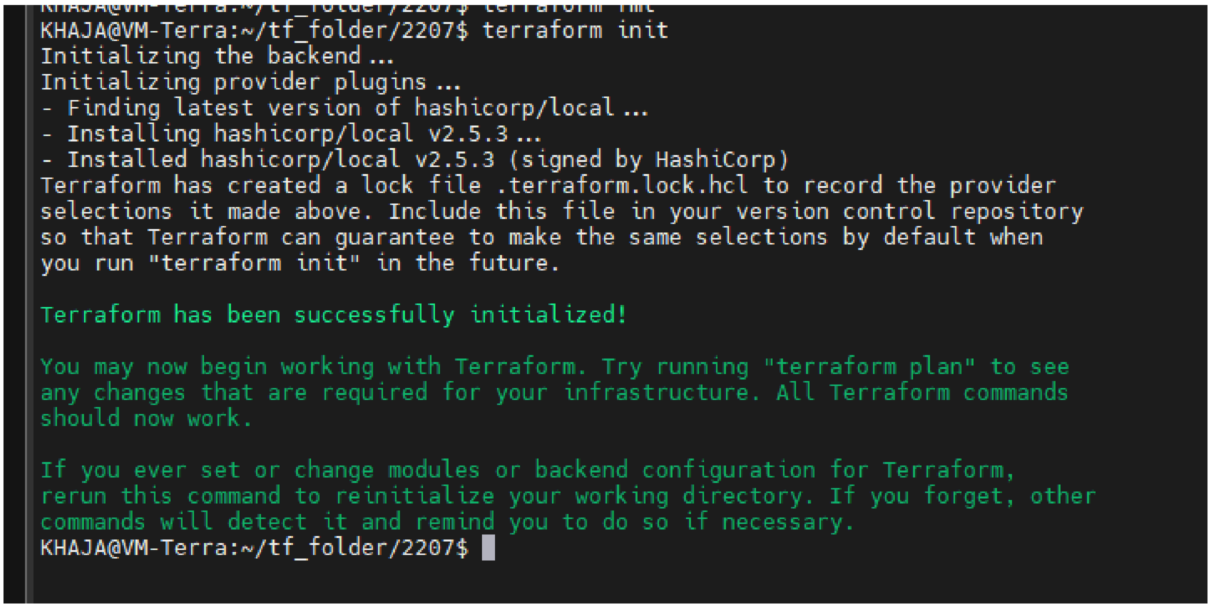
```
resource "local_file" "f1" {
  filename = var.filename1
  content  = var.content
}
resource "local_file" "f2" {
  filename = var.filename2
  content  = var.content
}
resource "local_file" "f3" {
  filename = var.filename3
  content  = var.content
}
resource "local_file" "f4" {
  filename = var.filename4
  content  = var.content
}
```



```
resource "local_file" "f1" {
  filename = var.filename1
  content  = var.content
}
resource "local_file" "f2" {
  filename = var.filename2
  content  = var.content
}
resource "local_file" "f3" {
  filename = var.filename3
  content  = var.content
}
resource "local_file" "f4" {
  filename = var.filename4
  content  = var.content
}

~
~
~
```

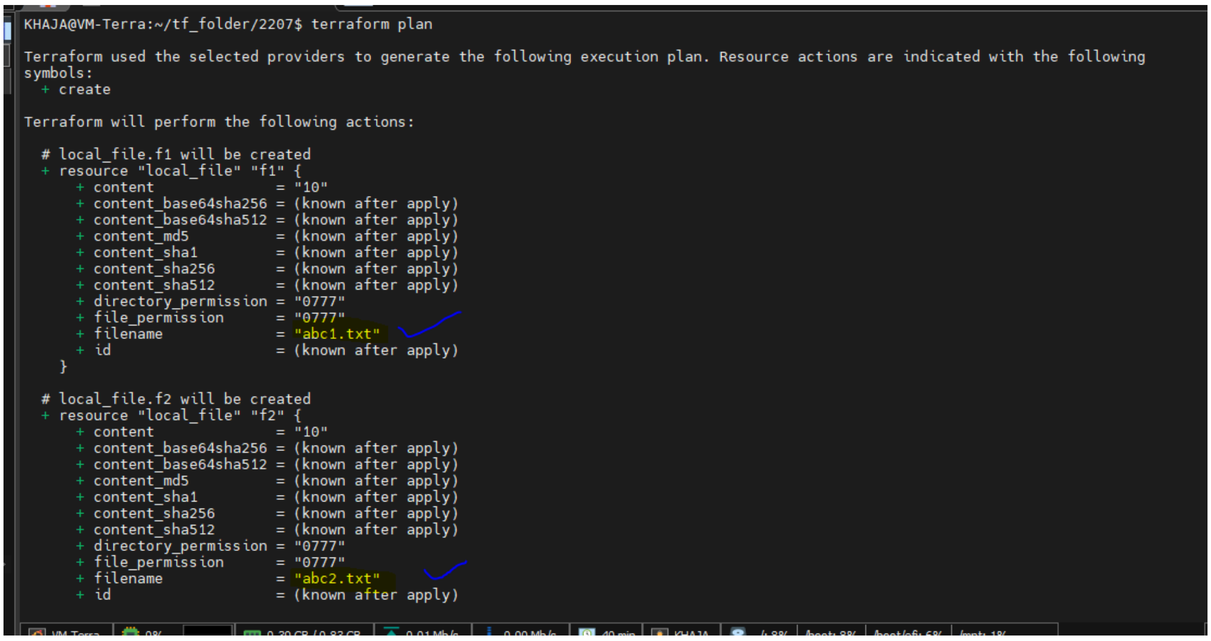
Variable Name	**Type**	**Default Value**	**Usage (as filename)**
filename1	string	"abc1.txt"	Directly used
filename2	string	"abc2.txt"	Directly used
filename3	bool	true	Converted to "true.txt"
filename4	number	15	Converted to "15.txt"
content	any	10	Can be any type, here a number

Execute the command terraform init

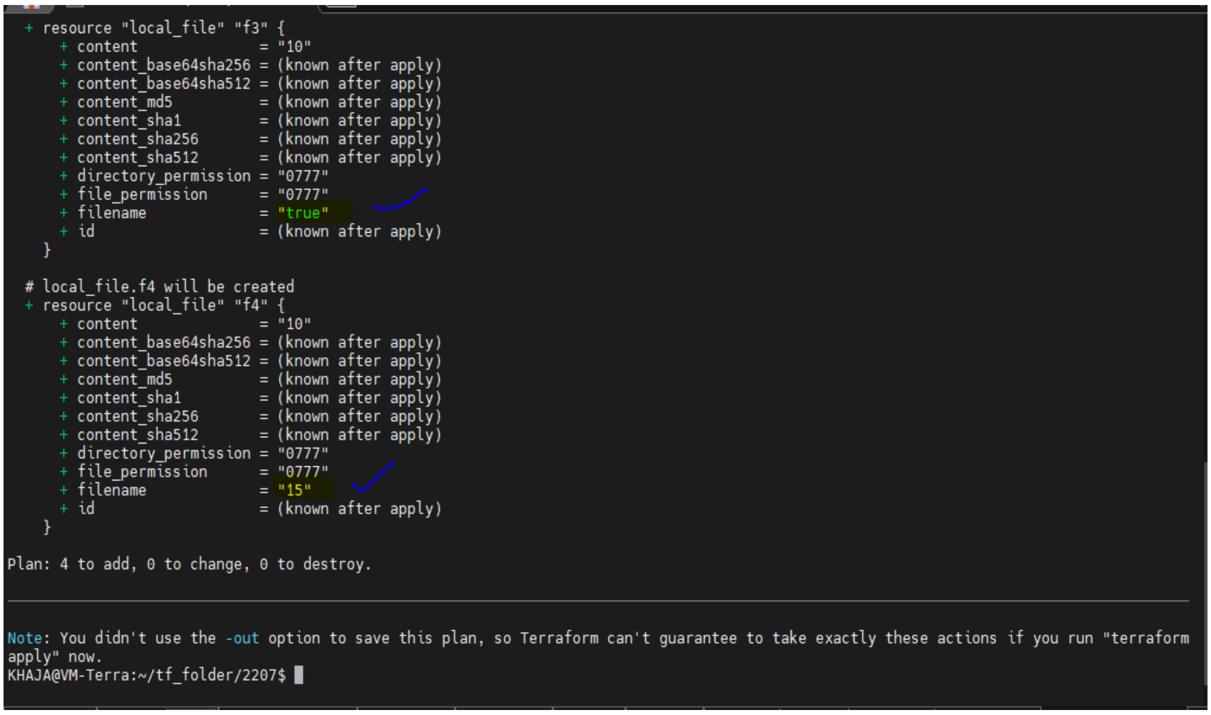


```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/2207$
```

Execute terraform plan

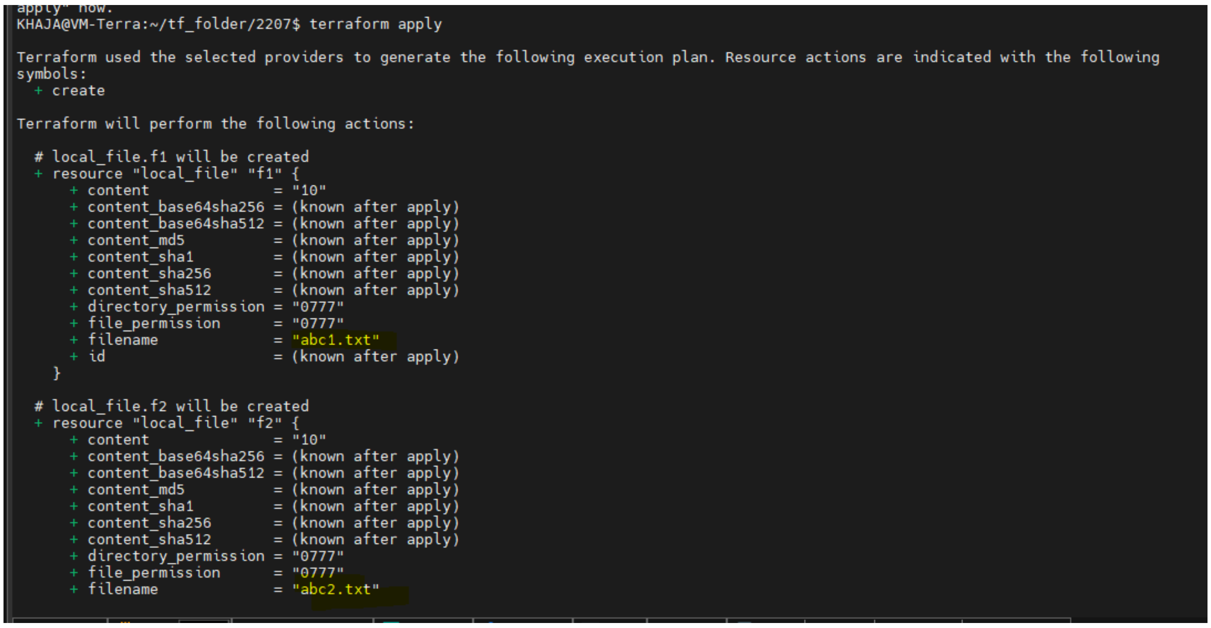


```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.f1 will be created
  + resource "local_file" "f1" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc1.txt"
      + id                   = (known after apply)
    }

  # local_file.f2 will be created
  + resource "local_file" "f2" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc2.txt"
      + id                   = (known after apply)
```

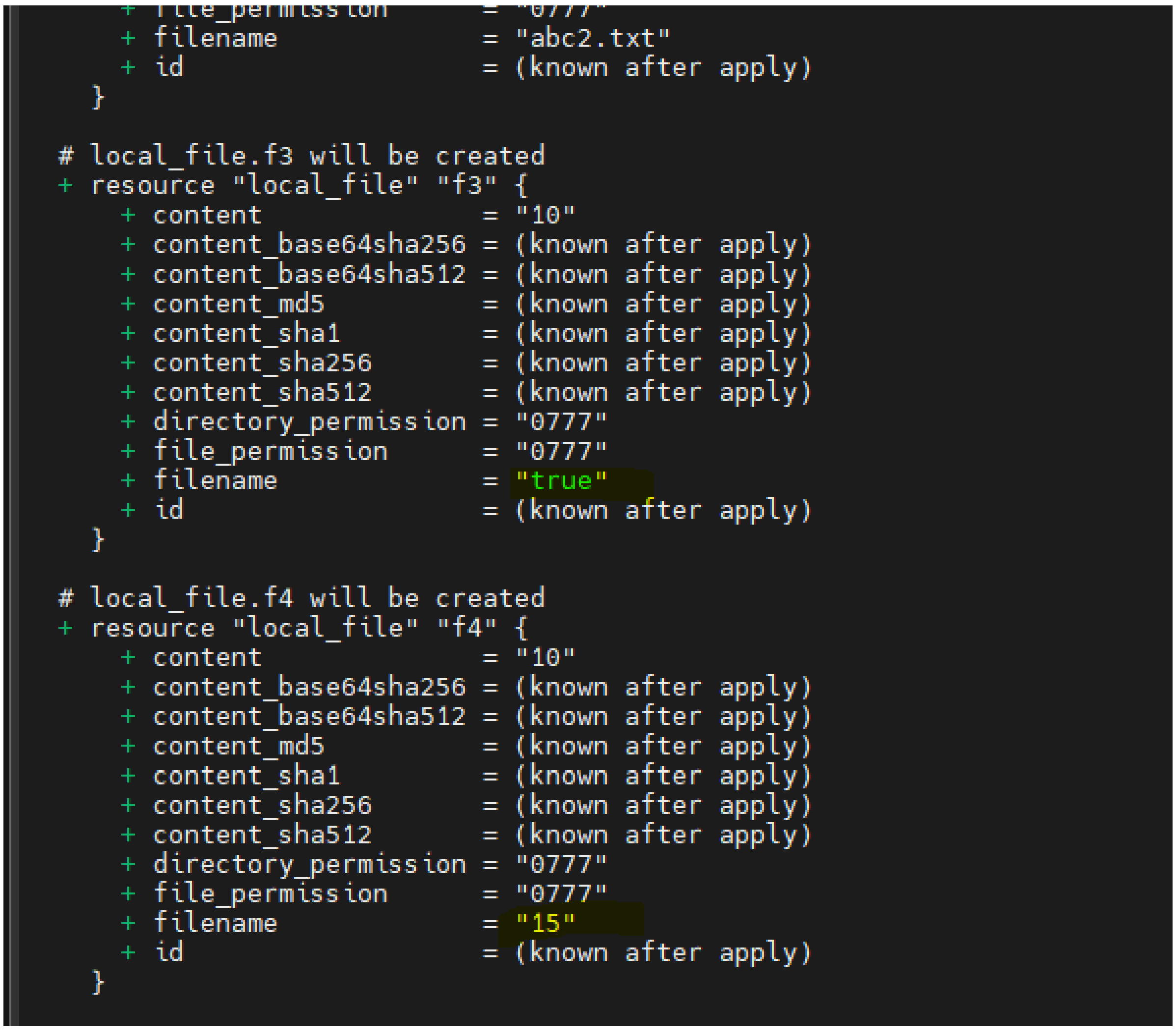


```
  + resource "local_file" "f3" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "true"
      + id                   = (known after apply)
    }

  # local_file.f4 will be created
  + resource "local_file" "f4" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "15"
      + id                   = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform
apply" now.
KHAJA@VM-Terra:~/tf_folder/2207$
```

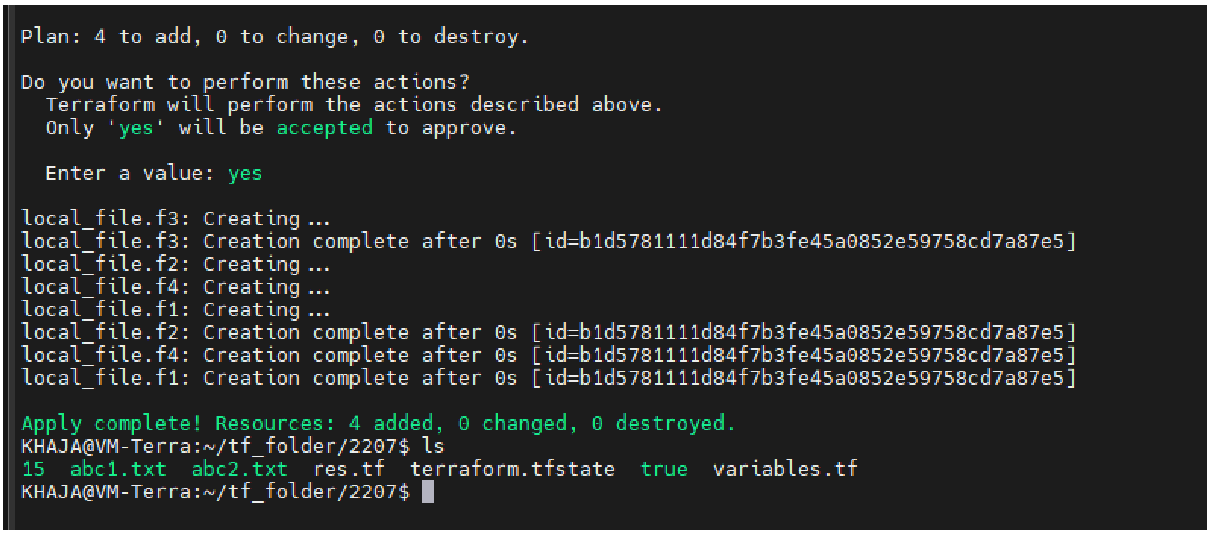
Execute terraform apply



```
apply" now.
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.f1 will be created
  + resource "local_file" "f1" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc1.txt"
      + id                   = (known after apply)
    }

  # local_file.f2 will be created
  + resource "local_file" "f2" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc2.txt"
```



```
      + file_permission      = "0777"
      + filename             = "abc2.txt"
      + id                   = (known after apply)
    }

  # local_file.f3 will be created
  + resource "local_file" "f3" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "true"
      + id                   = (known after apply)
    }

  # local_file.f4 will be created
  + resource "local_file" "f4" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "15"
      + id                   = (known after apply)
    }
```



```
Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.f3: Creating...
local_file.f3: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Creating...
local_file.f4: Creating...
local_file.f1: Creating...
local_file.f2: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$ ls
15  abc1.txt  abc2.txt  res.tf  terraform.tfstate  true  variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$
```

Here we can see four resources which we have given is created

Terraform Complex / Composite / Advanced Data Types:

1. List

An ordered sequence of values of the same or different type (depending on how you define it).

Syntax & Examples:

- Mixed-type list (type = list):

```
variable "mixed_list" {
  type    = list(any)  # default for 'list' without constraint
  default = ["test", 123, true, "test", 123]
}
```

- Homogeneous number list (type = list(number)):

```
variable "num_list" {
  type    = list(number)
  default = [1, 2, 3, 4, 5, 2, 4, 7, 1, 2]
}
```

- Nested list (list of lists):

```
variable "nested_list" {
  type    = list(list(number))
  default = [[1, 2], [3, 4], [5, 6]]
}
```

Access Elements:

```
var.varname[index]
var.num_list[2]  # returns 3
```

Error Scenario:

```
default = [1, 2]
var.num_list[2]  # Error: index out of range
```

tfvars Injection:

```
varname = [1, 2, 5, 6.7]
num_list = [1, 2, 5, 6.7]
```

2. Set

An unordered collection of unique values of the same type. Duplicate values are automatically removed.

Syntax & Example:

```
variable "unique_set" {
  type    = set(number)
  default = [1, 2, 3, 4, 5, 2, 4, 7, 1, 2]
}
# Interpreted as: [1, 2, 3, 4, 5, 7]
```

Notes:

- Indexing like var.unique_set[0] may not work reliably since set is unordered.
- Treat set as similar to a de-duped list, but avoid accessing via index.

3. Map

A key-value pair data structure with keys as strings.

Basic map (type = map):

```
variable "user_info" {
  type = map(any)
  default = {
    name     = "adi"
    id       = 123
    isactive = true
  }
}
```

Typed map (type = map(string)):

```
variable "user_string_map" {
  type = map(string)
  default = {
    name     = "adi"
    id       = "123"
    isactive = "yes"
  }
}
```

Number map:

```
variable "numeric_map" {
  type = map(number)
  default = {
    id    = 12345
    phone = 43154431
  }
}
```

Map of list:

```
variable "map_list" {
  type = map(list(string))
  default = {
    devs  = ["ram", "raj"]
    admins = ["khaja", "admin"]
  }
}
```

Access:

```
var.user_info["id"]
var.user_info.id
```

Error Scenario:

```
var.user_info.phoneno  # Key doesn't exist → error
```

tfvars Injection:

example : varname = { name = "test", dob = 123 }

```
 user_info = {
  name = "test"
  dob  = 123
}
```

4. Tuple

A fixed-length, ordered list with elements of different types. Each element has a specific position and type.

Syntax:

```
variable "my_tuple" {
  type = tuple([string, number, bool])
  default = ["hello", 10, true]
}
```

Access:

```
var.my_tuple[0]  # "hello"
var.my_tuple[2]  # true
```

Error:

- If you pass wrong type or wrong length → validation error.
- Example:

```
default = ["test", "wrong", false]  # Second element must be number
```

5. Object

A custom user-defined structure with fixed attributes and types — think of it as a strict version of a map.

Syntax:

```
variable "employee" {
  type = object({
    name     = string
    id       = number
    isactive = bool
  })

  default = {
    name     = "Khaja"
    id       = 101
    isactive = true
  }
}
```

Access:

```
var.employee.name     # "Khaja"
var.employee.isactive # true
```

Error Scenario:

```
var.employee.email  # Attribute not defined → error
```

creating of resources :

# creating resource using set type

```
resource "local_file" "f5" {
  filename = var.filename5[0] # (here its using via index)
  content  = var.content
}
resource "local_file" "f6" {
  filename = var.filename6[2]
  content  = var.content
}
```

# creating resource using map type

```
resource "local_file" "f7" {
  filename = var.filename7.name # (here its using via key name)
  content  = var.content
}
resource "local_file" "f8" {
  filename = var.filename8.id
  content  = var.content
}
```

For Variable Declaration:

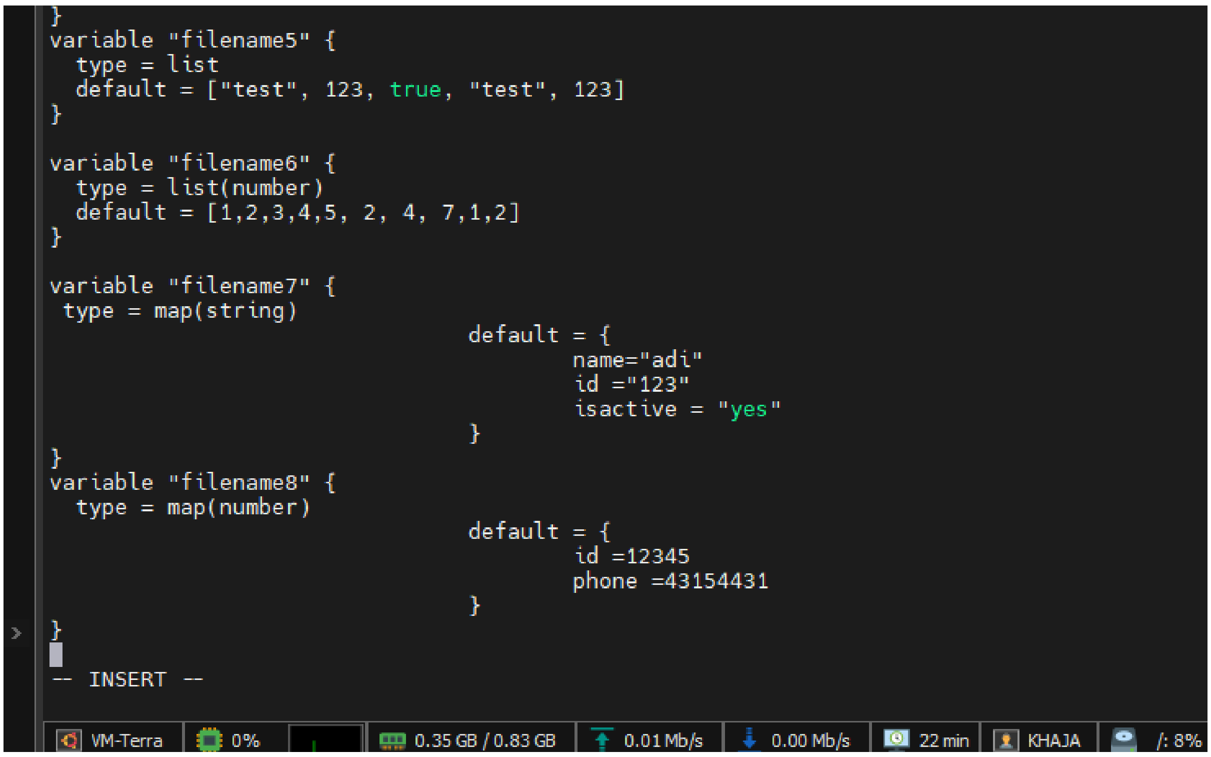
# list

```
variable "filename5" {
  type = list
  default = ["test", 123, true, "test", 123]
}

variable "filename6" {
  type = list(number)
  default = [1,2,3,4,5, 2, 4, 7,1,2]
}
```

# For Map

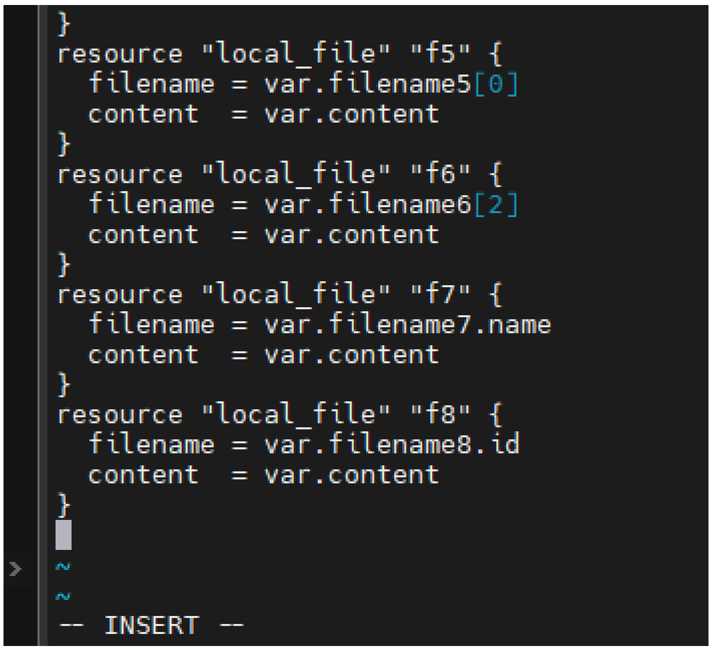
```
variable "filename7" {
 type = map(string)
          default = {
          name="adi"
          id ="123"
          isactive = "yes"
                    }
}
variable "filename8" {
  type = map(number)
                                        default = {
                                                  id =12345
                                                  phone =43154431
                                        }
}
```

Variable are mapped to variable file like vi variable.tf



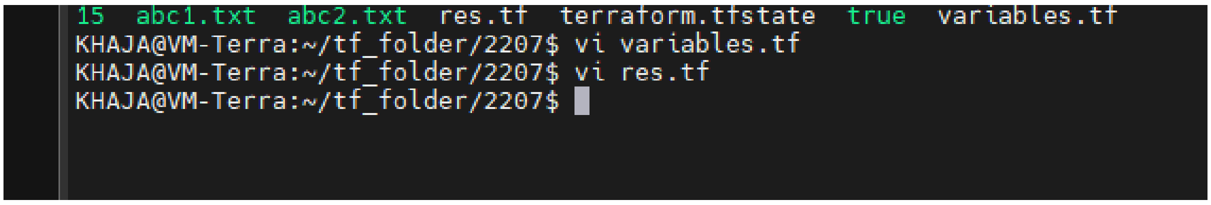
```
}
variable "filename5" {
  type = list
  default = ["test", 123, true, "test", 123]
}

variable "filename6" {
  type = list(number)
  default = [1,2,3,4,5, 2, 4, 7,1,2]
}

variable "filename7" {
 type = map(string)
                                default = {
                                        name="adi"
                                        id ="123"
                                        isactive = "yes"
                                }
}
variable "filename8" {
  type = map(number)
                                default = {
                                        id =12345
                                        phone =43154431
                                }
}

-- INSERT --
```

Adding resources to the resource file “vi res.tf”



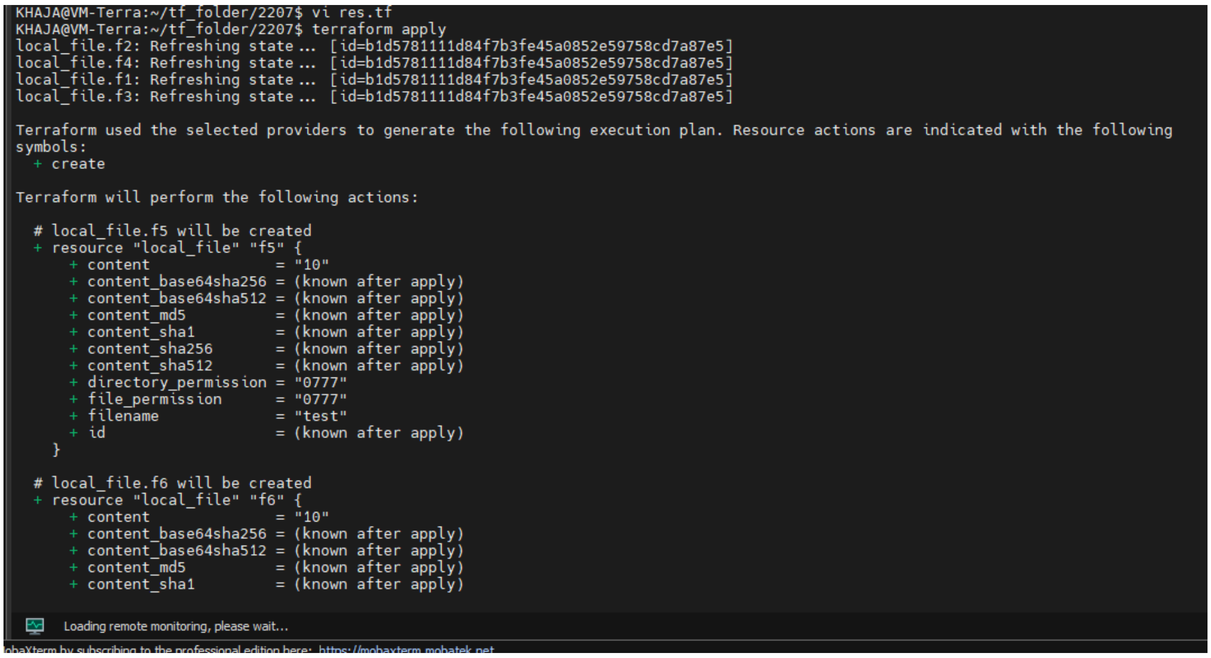
```
}
resource "local_file" "f5" {
  filename = var.filename5[0]
  content  = var.content
}
resource "local_file" "f6" {
  filename = var.filename6[2]
  content  = var.content
}
resource "local_file" "f7" {
  filename = var.filename7.name
  content  = var.content
}
resource "local_file" "f8" {
  filename = var.filename8.id
  content  = var.content
}

~
~
-- INSERT --
```

Know we have variable and resources with respect to their files



```
15  abc1.txt  abc2.txt  res.tf  terraform.tfstate  true  variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$ vi variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$ vi res.tf
KHAJA@VM-Terra:~/tf_folder/2207$
```

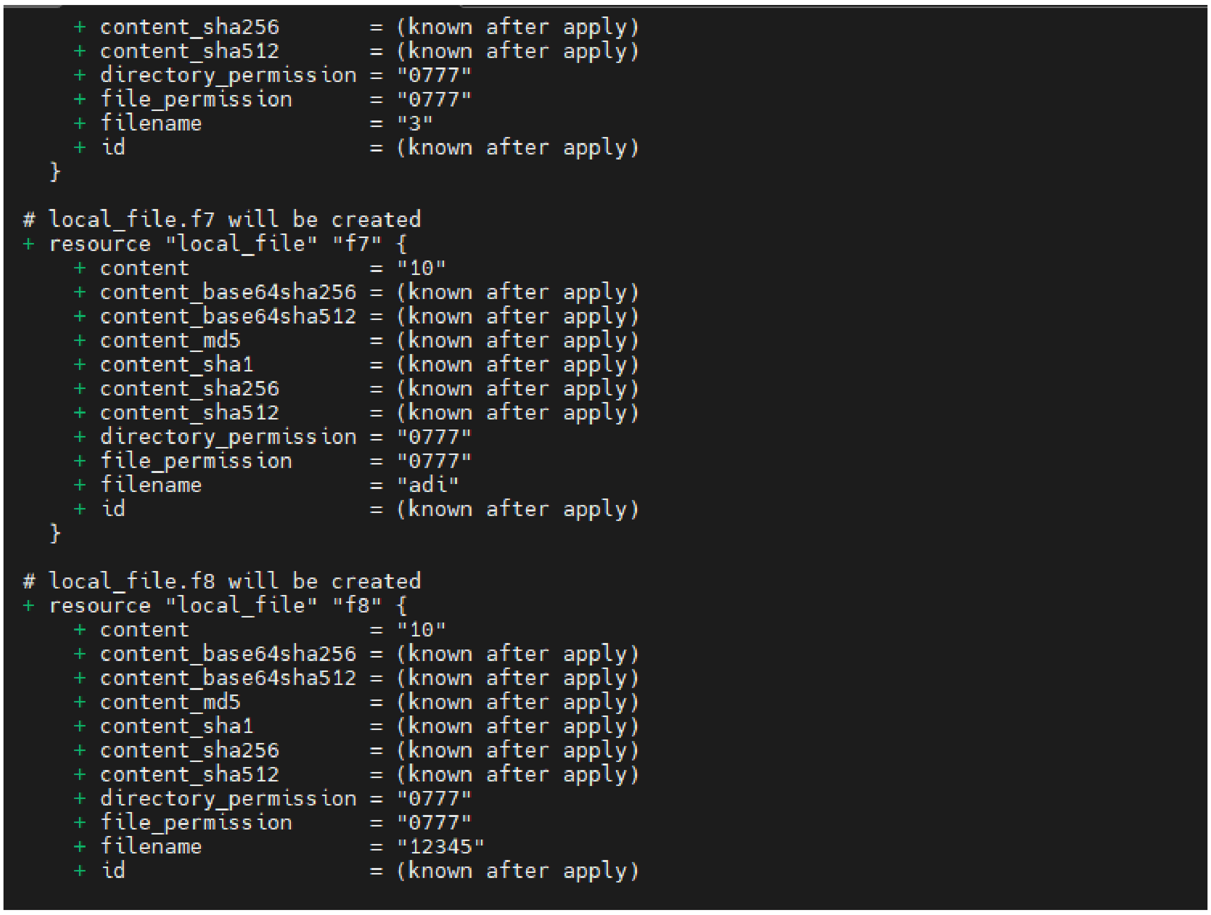
Execute the command apply



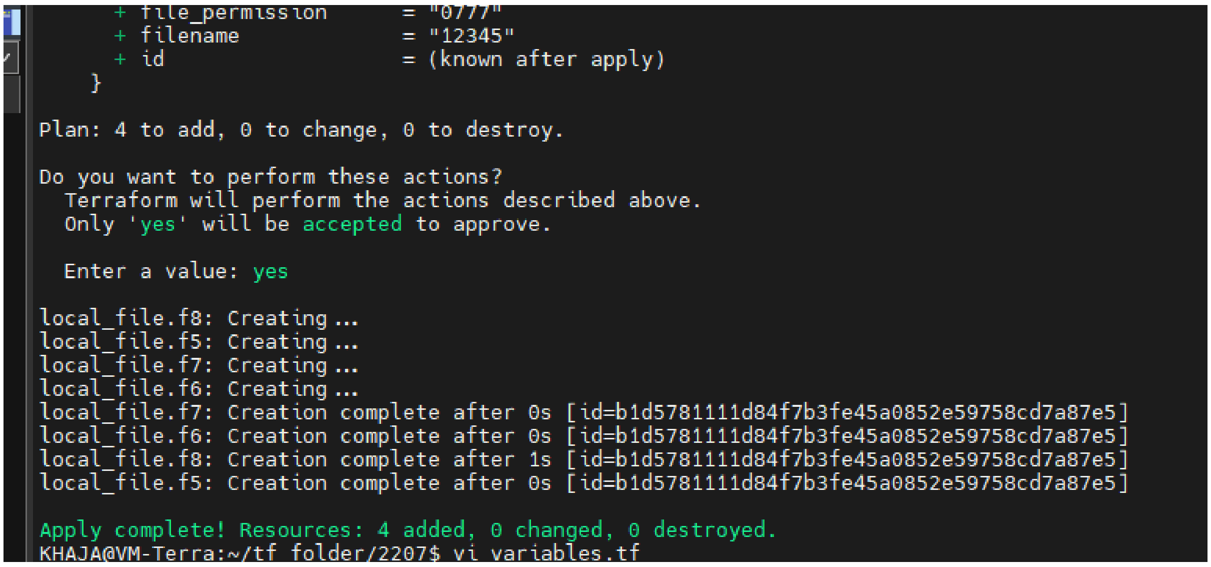
```
KHAJA@VM-Terra:~/tf_folder/2207$ vi res.tf
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.f5 will be created
  + resource "local_file" "f5" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "test"
      + id                   = (known after apply)
    }

  # local_file.f6 will be created
  + resource "local_file" "f6" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
```

Loading remote monitoring, please wait...

obaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



```
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "3"
      + id                   = (known after apply)
    }

  # local_file.f7 will be created
  + resource "local_file" "f7" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "adi"
      + id                   = (known after apply)
    }

  # local_file.f8 will be created
  + resource "local_file" "f8" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "12345"
      + id                   = (known after apply)
```



```
      + file_permission      = "0777"
      + filename             = "12345"
      + id                   = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.f8: Creating...
local_file.f5: Creating...
local_file.f7: Creating...
local_file.f6: Creating...
local_file.f7: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Creation complete after 1s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$ vi variables.tf
```

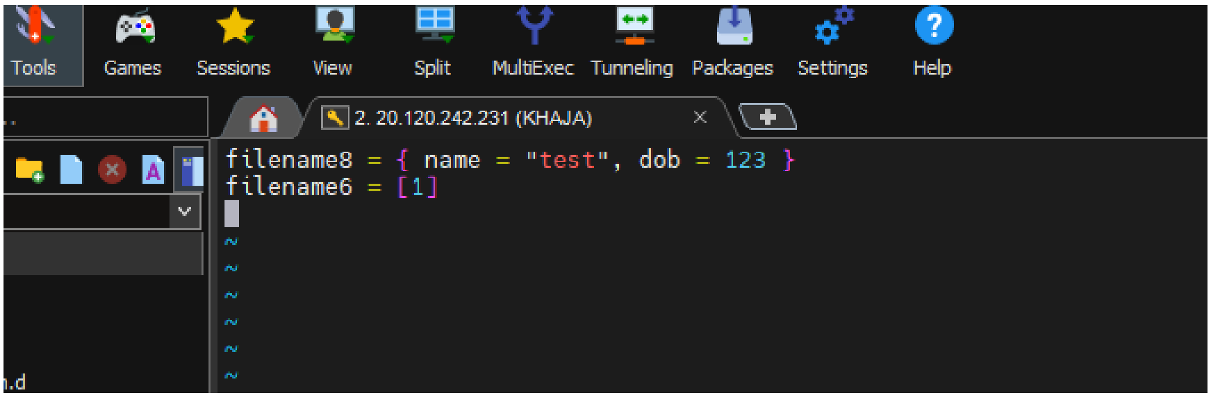
Inject the values by using .tfvar

vi terraform.tfvar

lets inject two values

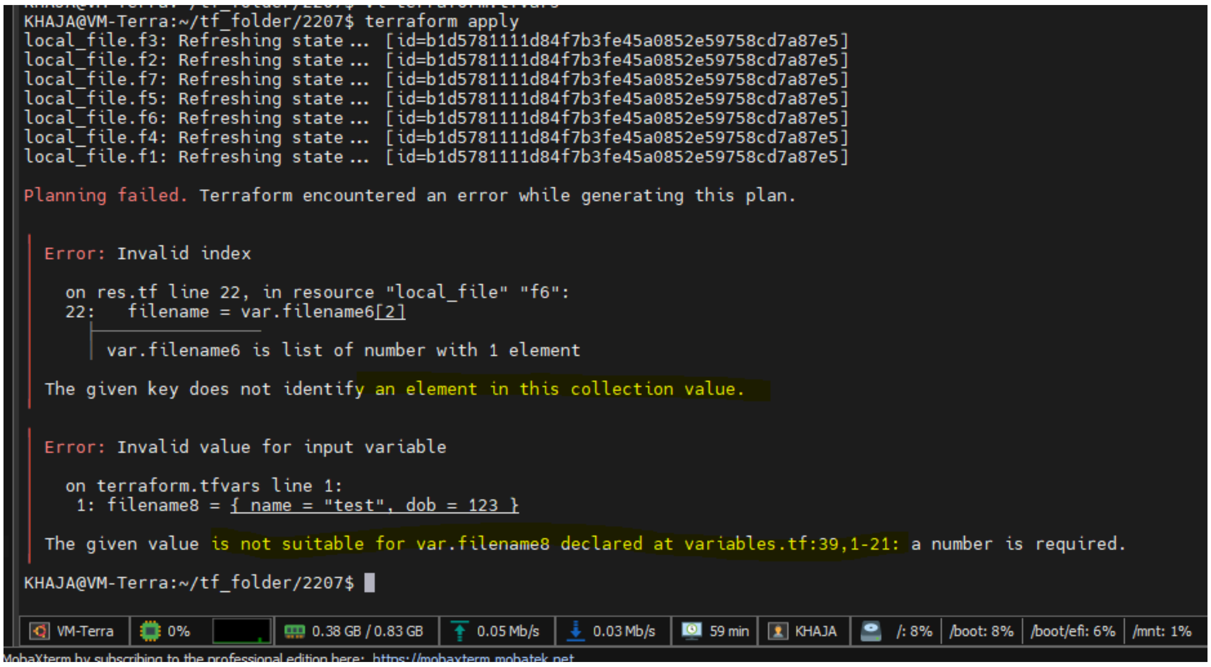
filename8 = { name = "test", dob = 123 }

filename6 = [1]   # we are injecting one value



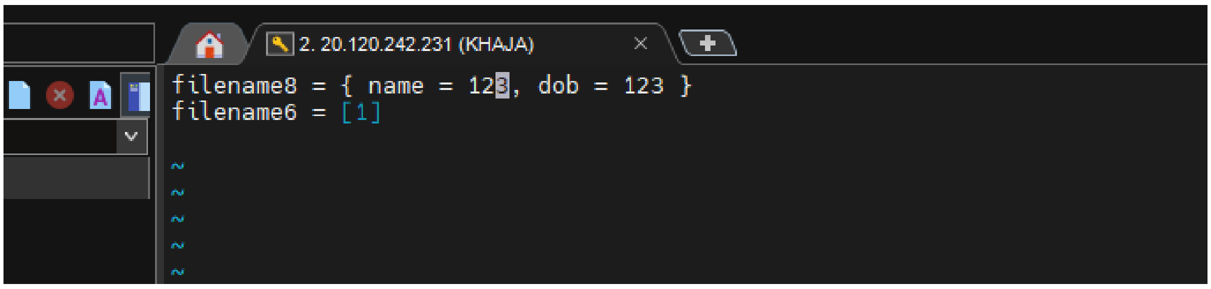
```
filename8 = { name = "test", dob = 123 }
filename6 = [1]
```

When we execute terraform apply



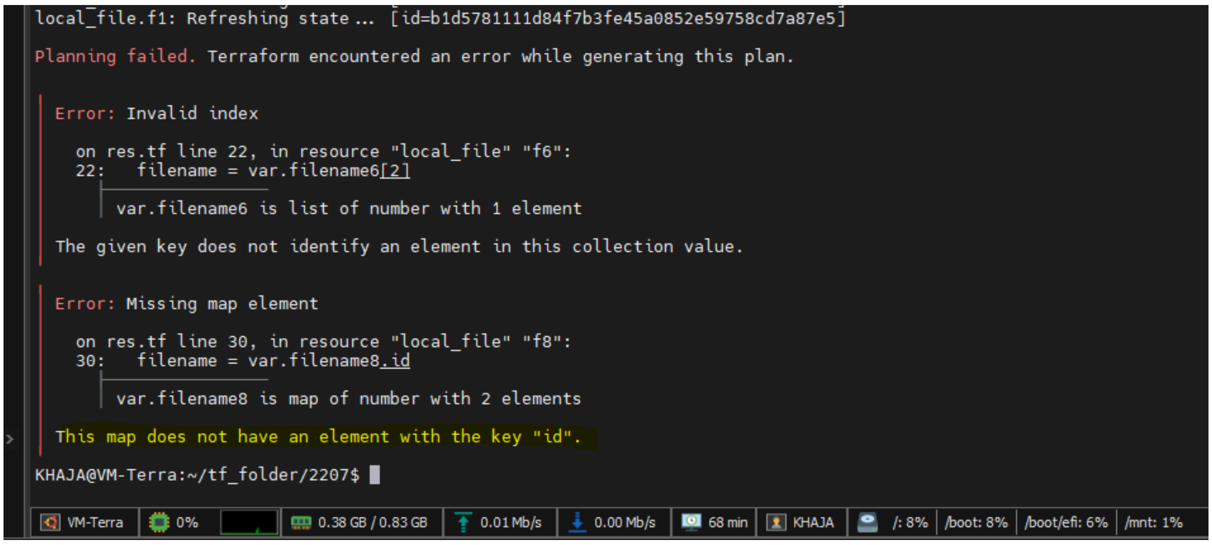
```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Planning failed. Terraform encountered an error while generating this plan.

│ Error: Invalid index
│
│   on res.tf line 22, in resource "local_file" "f6":
│   22:     filename = var.filename6[2]
│     ├────────────────
│     │ var.filename6 is list of number with 1 element
│
│ The given key does not identify an element in this collection value.
│
│ Error: Invalid value for input variable
│
│   on terraform.tfvars line 1:
│    1: filename8 = { name = "test", dob = 123 }
│
│ The given value is not suitable for var.filename8 declared at variables.tf:39,1-21: a number is required.

KHAJA@VM-Terra:~/tf_folder/2207$
```

- It show an error because here filename8 we use number
- In filename6 given values is [2] but here we have only one element



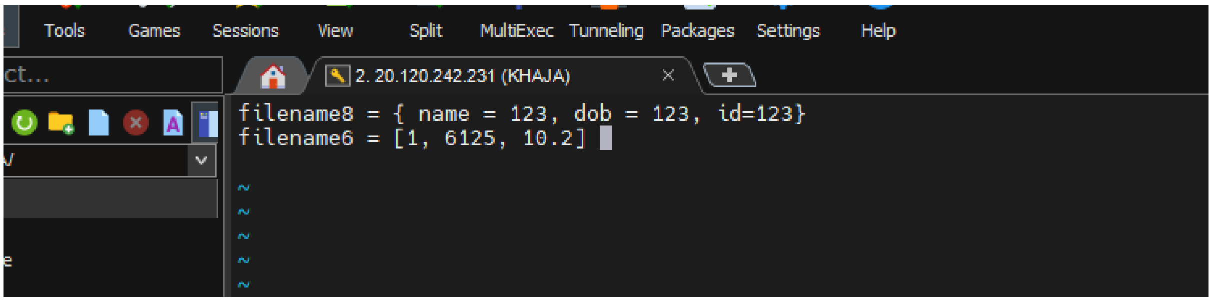
```
filename8 = { name = 123, dob = 123 }
filename6 = [1]
```

Here I change name=123 it's a number after apply



```
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Planning failed. Terraform encountered an error while generating this plan.

│ Error: Invalid index
│
│   on res.tf line 22, in resource "local_file" "f6":
│   22:   filename = var.filename6[2]
│     ├────────────────
│     │ var.filename6 is list of number with 1 element
│
│ The given key does not identify an element in this collection value.

│ Error: Missing map element
│
│   on res.tf line 30, in resource "local_file" "f8":
│   30:   filename = var.filename8.id
│     ├────────────────
│     │ var.filename8 is map of number with 2 elements
│
│ This map does not have an element with the key "id".

KHAJA@VM-Terra:~/tf_folder/2207$
```

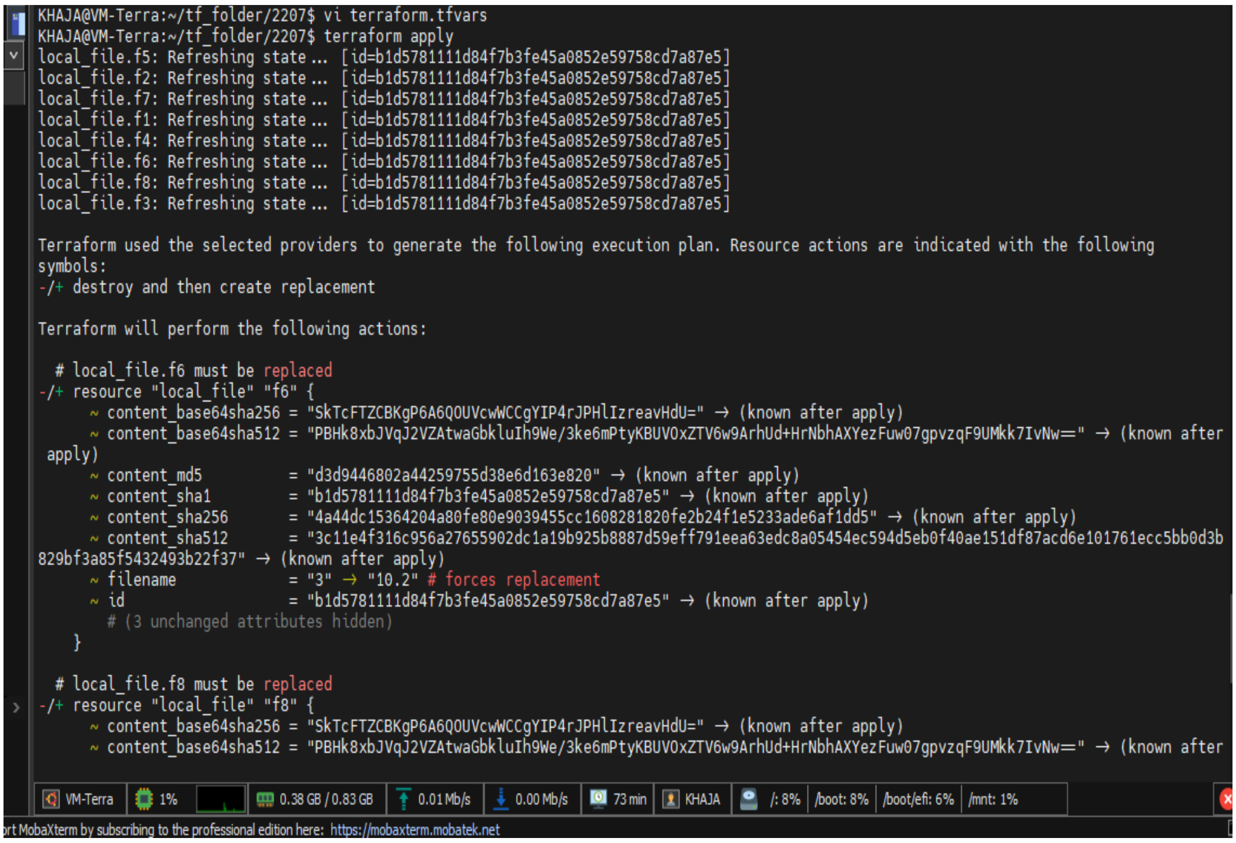
Again its show me an error does not have an element



```
filename8 = { name = 123, dob = 123, id=123}
filename6 = [1, 6125, 10.2]
~
~
~
~
~
```

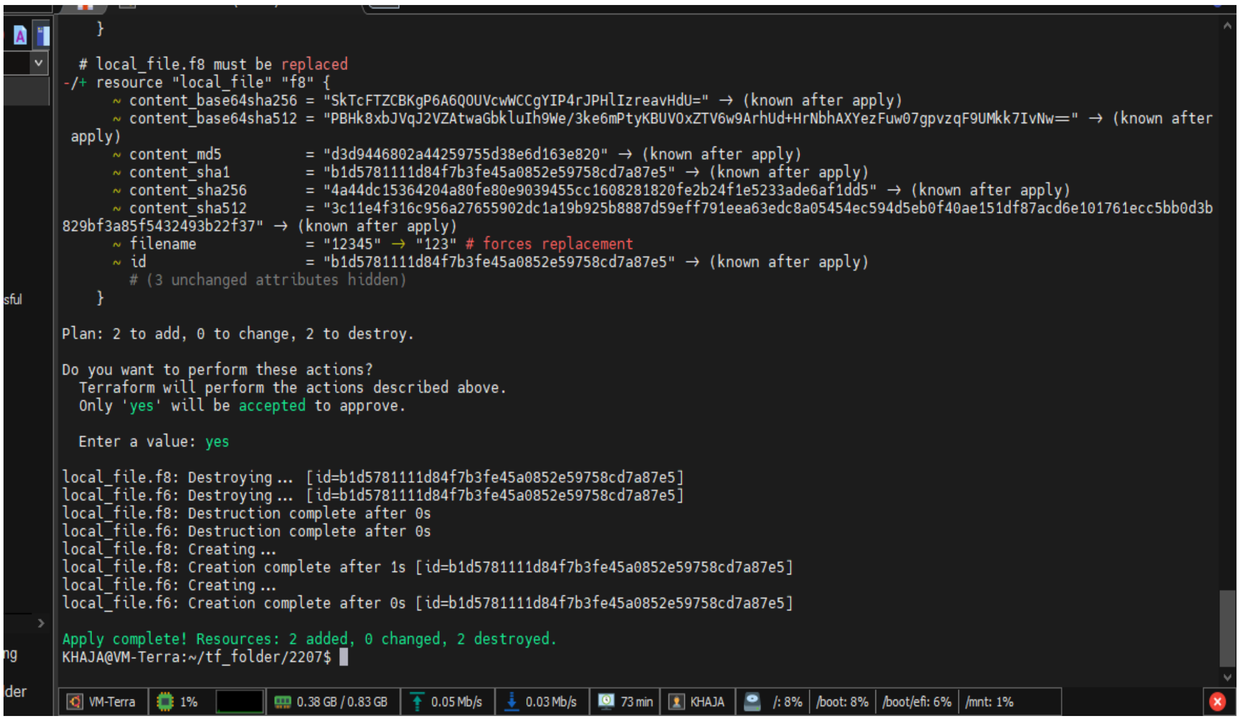
Know I have change the value id =123 a random number

Execute terraform apply know



```
KHAJA@VM-Terra:~/tf_folder/2207$ vi terraform.tfvars
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f5: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # local_file.f6 must be replaced
-/+ resource "local_file" "f6" {
      ~ content_base64sha256 = "SkTcFTZCBKgP6A6QOUVcwWCCgYIP4rJPHlIzreavHdU=" -> (known after apply)
      ~ content_base64sha512 = "PBHk8xbJVqJ2VZAtwaGbkluIh9We/3ke6mPtyKBUVOxZTV6w9ArhUd+HrNbhAXYezFuw07gpvzqF9UMkk7IvNw==" -> (known after
 apply)
      ~ content_md5          = "d3d9446802a44259755d38e6d163e820" -> (known after apply)
      ~ content_sha1         = "b1d5781111d84f7b3fe45a0852e59758cd7a87e5" -> (known after apply)
      ~ content_sha256       = "4a44dc15364204a80fe80e9039455cc1608281820fe2b24f1e5233ade6af1dd5" -> (known after apply)
      ~ content_sha512       = "3c11e4f316c956a27655902dc1a19b925b8887d59eff791eea63edc8a05454ec594d5eb0f40ae151df87acd6e101761ecc5bb0d3b
829bf3a85f5432493b22f37" -> (known after apply)
      ~ filename             = "3" -> "10.2" # forces replacement
      ~ id                   = "b1d5781111d84f7b3fe45a0852e59758cd7a87e5" -> (known after apply)
        # (3 unchanged attributes hidden)
    }

  # local_file.f8 must be replaced
-/+ resource "local_file" "f8" {
      ~ content_base64sha256 = "SkTcFTZCBKgP6A6QOUVcwWCCgYIP4rJPHlIzreavHdU=" -> (known after apply)
      ~ content_base64sha512 = "PBHk8xbJVqJ2VZAtwaGbkluIh9We/3ke6mPtyKBUVOxZTV6w9ArhUd+HrNbhAXYezFuw07gpvzqF9UMkk7IvNw==" -> (known after
```

VM-Terra
1%
0.38 GB / 0.83 GB
0.01 Mb/s
0.00 Mb/s
73 min
KHAJA
/: 8%
/boot: 8%
/boot/efi: 6%
/mnt: 1%
rt MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

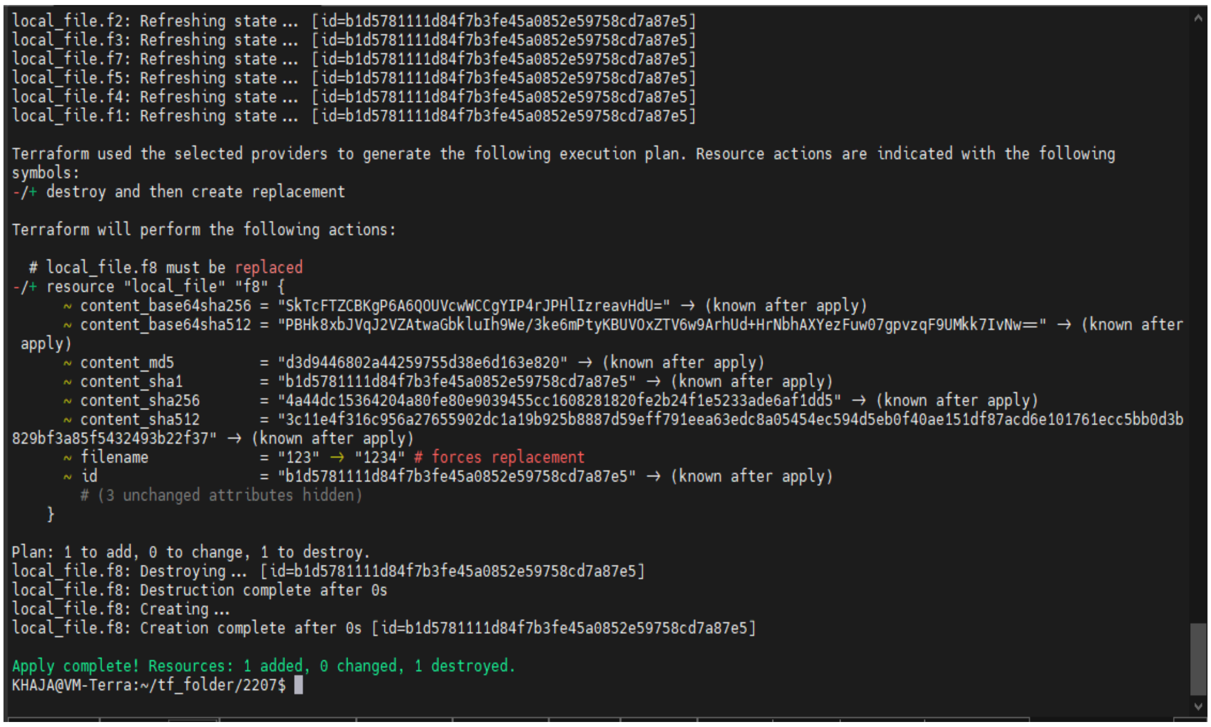


```
    }

  # local_file.f8 must be replaced
-/+ resource "local_file" "f8" {
      ~ content_base64sha256 = "SkTcFTZCBKgP6A6QOUVcwWCCgYIP4rJPHlIzreavHdU=" → (known after apply)
      ~ content_base64sha512 = "PBHk8xbJVqJ2VZAtwaGbkluIh9We/3ke6mPtyKBUVOxZTV6w9ArhUd+HrNbhAXYezFuw07gpvzqF9UMkk7IvNw==" → (known after
 apply)
      ~ content_md5          = "d3d9446802a44259755d38e6d163e820" → (known after apply)
      ~ content_sha1         = "b1d5781111d84f7b3fe45a0852e59758cd7a87e5" → (known after apply)
      ~ content_sha256       = "4a44dc15364204a80fe80e9039455cc1608281820fe2b24f1e5233ade6af1dd5" → (known after apply)
      ~ content_sha512       = "3c11e4f316c956a27655902dc1a19b925b8887d59eff791eea63edc8a05454ec594d5eb0f40ae151df87acd6e101761ecc5bb0d3b
829bf3a85f5432493b22f37" → (known after apply)
      ~ filename             = "12345" → "123" # forces replacement
      ~ id                   = "b1d5781111d84f7b3fe45a0852e59758cd7a87e5" → (known after apply)
        # (3 unchanged attributes hidden)
    }

Plan: 2 to add, 0 to change, 2 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.f8: Destroying... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Destroying... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Destruction complete after 0s
local_file.f6: Destruction complete after 0s
local_file.f8: Creating...
local_file.f8: Creation complete after 1s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Creating...
local_file.f6: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Apply complete! Resources: 2 added, 0 changed, 2 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$ 
```

VM-Terra | 1% | 0.38 GB / 0.83 GB | 0.05 Mb/s | 0.03 Mb/s | 73 min | KHAJA | /: 8% | /boot: 8% | /boot/efi: 6% | /mnt: 1%

We can also terraform apply -auto-approve

Automatically **applies the Terraform plan** without prompting for **manual confirmation**.



```
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # local_file.f8 must be replaced
-/+ resource "local_file" "f8" {
      ~ content_base64sha256 = "SkTcFTZCBKgP6A6QOUVcwWCCgYIP4rJPHlIzreavHdU=" → (known after apply)
      ~ content_base64sha512 = "PBHk8xbJVqJ2VZAtwaGbkluIh9We/3ke6mPtyKBUVOxZTV6w9ArhUd+HrNbhAXYezFuw07gpvzqF9UMkk7IvNw==" → (known after
 apply)
      ~ content_md5          = "d3d9446802a44259755d38e6d163e820" → (known after apply)
      ~ content_sha1         = "b1d5781111d84f7b3fe45a0852e59758cd7a87e5" → (known after apply)
      ~ content_sha256       = "4a44dc15364204a80fe80e9039455cc1608281820fe2b24f1e5233ade6af1dd5" → (known after apply)
      ~ content_sha512       = "3c11e4f316c956a27655902dc1a19b925b8887d59eff791eea63edc8a05454ec594d5eb0f40ae151df87acd6e101761ecc5bb0d3b
829bf3a85f5432493b22f37" → (known after apply)
      ~ filename             = "123" → "1234" # forces replacement
      ~ id                   = "b1d5781111d84f7b3fe45a0852e59758cd7a87e5" → (known after apply)
        # (3 unchanged attributes hidden)
    }

Plan: 1 to add, 0 to change, 1 to destroy.
local_file.f8: Destroying... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Destruction complete after 0s
local_file.f8: Creating...
local_file.f8: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$ 
```